AMENDMENTS TO THE SPECIFICATION:

Please amend the specification as follows:

Please replace paragraphs [001], [013], [032], [037] - [039], [055], [056], and [060] - [062] with the following paragraphs:

[001] U.S. Patent Application Serial No. 10/035,747, filed on even date herewith in the name of Guy L. Steele Jr. and entitled "Floating Point Unit in which Floating Point Status Information is Encoded in Floating Point Representations," assigned to the assignee of the present application, is hereby incorporated by reference.

[013] There are several drawbacks to the conventional approach. First, setting the 32-bit integer represented by the least significant fraction field bits equal to zero produces a less accurate high part. In particular, it causes the difference between the fraction of the resulting floating point number and the fraction of the input floating point number to undesirably become too large. Second, storing the 64-bit input floating point number to memory typically requires more time than an operation performed in registers on most computers with their associated programming language compilers. Third, sensible results are not always obtained when the input floating point number is in a denormalized, NaN, infinity, or overflow format according to the floating-point number system disclosed in related U.S. Patent Application Serial No. 10/035,747 or in an infinity or NaN format according to IEEE Std. 754. For example, typically, to compute the "low part" of a number, the "high part" of the number is subtracted from the number. However, if the number is infinity, the high part of the number will also be infinity using

the conventional approach. The subtraction of the high part of the number from the number will produce a floating point number having the NaN format. Also, clearing the low bits of a NaN formatted operand may lose valuable information. Therefore, inputs in a denormalized, NaN, infinity, or overflow format according to the floating-point number system disclosed in related U.S. Patent Application Serial No. 10/035,747 or in an infinity or NaN format according to IEEE Std. 754 must be tested for and handled separately as special cases in the conventional approach, which introduces additional undesirable costs in memory space and time.

[032] Fig. 1 illustrates an exemplary floating point unit 100 that may be used to compute the high part of a floating point number. That is, the floating point unit 100 receives an input floating point number and outputs a resulting floating point number that may have a smaller fraction than the input floating point number and whose significand may be squared exactly, without loss of precision. The floating point unit 100 computes the high part of a floating point number based on the format of the floating point number. The floating point unit 100 computes the high part of special input floating point numbers that are in the overflow, infinity, or NaN format shown in Fig. 12 and also described in related U.S. Patent Application Serial No. 10/035,747 or in the infinity or NaN format according to IEEE Std. 754 so that the low part of these special floating point numbers may be computed to produce sensible results. The floating point unit 100 may adaptively compute the high part of an input floating point number that is in a denormalized format according to IEEE Std. 754. That is, floating point unit 100 does not simply clear a predetermined number of least significant fraction field bits of the input floating point number. Instead, the number of least significant

fraction field bits that may be cleared by floating point unit 100 depends on the number of leading zeros in the fraction field of the input floating point number.

[037] The result generator 140 receives the floating point operand 200 from the operand buffer 110, the signals indicating the operand's format from the operand analysis circuit 120, and the resulting fraction field low part provided by the processing circuit 130 to assemble a resulting floating point number equivalent to the high part of the floating point operand 200 stored in operand buffer 110. In a more detailed description of an embodiment of the invention, the resulting floating point number assembled by result generator 140 may have the same sign as the floating point operand 200 stored in operand buffer 110. Furthermore, the resulting floating point number assembled by result generator 140 may have the same exponent field 220 as the floating point operand 200 stored in operand buffer 110, unless the floating point operand 200 is in an infinity or overflow format shown in Fig. 12 and in related U.S. Patent Application Serial No. 10/035,747 or an infinity format according to IEEE Std. 754, as discussed in greater detail below. Still further, the resulting floating point number assembled by result generator 140 may have the same fraction field high part 230 as the floating point operand 200 stored in operand buffer 110, unless the floating point operand 200 is in an infinity or overflow format shown in Fig. 12 and in related U.S. Patent Application Serial No. 10/035,747 or an infinity format according to IEEE Std. 754, as discussed in greater detail below. Finally, the fraction field low part of the resulting floating point number assembled by result generator 140 may equal zero, unless the floating point operand 200 stored in operand buffer 110 is (1) in a NaN format according to U.S. Patent Application Serial No. 10/035,747 or IEEE Std. 754; (2)

a subprecise number in a delimited format according to U.S. Patent No. 6,131,106; or (3) a denormalized format according to IEEE Std. 754. If the floating point operand 200 is in a NaN format according to U.S. Patent Application Serial No. 10/035,747 or IEEE Std. 754, the fraction field low part of the resulting floating point number assembled by result generator 140 may equal the fraction field low part 240 of the operand 200. If the floating point operand 200 is a subprecise number in a delimited format according to U.S. Patent No. 6,131,106 or in a denormalized format according to IEEE Std. 754, the fraction field low part of the resulting floating point number assembled by result generator 140 may equal the fraction field low part provided by the processing circuit 130. The resulting floating point number assembled by result generator 140 may have a smaller fraction than the input floating point operand stored in operand buffer 110 and may have a significand that may be squared exactly, without loss of precision.

[038] Fig. 3 illustrates a first exemplary embodiment of the operand analysis circuit 120 that may be used when the floating point operand 200 stored in the operand buffer 110 is in a zero, underflow, overflow, infinity, or NaN format according to U.S. Patent Application Serial No. 10/035,747 or in a normalized delimited format as disclosed in U.S. Patent No. 6,131,106.

[039] Fig. 12 illustrates several exemplary formats for an operand, such as the zero format 1210, the underflow format 1220, the normalized nonzero format 1230, the overflow format 1240, the infinity format 1250, and the NaN format 1260, as disclosed in U.S. Patent Application Serial No. 10/035,747. As shown in the embodiment illustrated in Fig. 12, in the zero format 1210, the exponent field bits, $e_{msb} \cdots e_{lsb}$, and the fraction

field bits, $f_{msb} \cdots f_{lsb}$, are all binary zeros. In the underflow format 1220, the exponent field bits, $e_{msb} \cdots e_{lsb}$, are all binary zeros, the twenty-two most significant fraction field bits, $f_{msb} \cdots f_{lsb+1}$, are all binary zeros, and the least significant fraction field bit, f_{lsb}, is a binary one. In the normalized nonzero format 1230, the exponent field bits, $e_{msb} \cdots e_{lsb}$, are neither all binary ones nor all binary zeros. In the overflow format 1240, the seven most significant exponent field bits, $e_{msb} \cdots e_{lsb+1}$, are all binary ones, with the least significant bit, e_{lsb}, being a binary zero, and the fraction field bits, $f_{msb} \cdots f_{lsb}$, are all binary ones. In the infinity format 1250, the exponent field bits, $e_{msb} \cdots e_{lsb}$, are all binary ones, the eighteen most significant fraction field bits, $f_{msb} \cdots f_{lsb+5}$, are all binary zeros, and the five least significant fraction field bits, $f_{lsb+4} \cdots f_{lsb}$, are flags. In the NaN format 1260, the exponent field bits, $e_{msb} \cdots e_{lsb}$, are all binary ones, the eighteen most significant fraction field bits, $f_{msb} \cdots f_{lsb+5}$, are not all binary zeros, and the five least significant fraction field bits, $f_{lsb+4} \cdots f_{lsb}$, are flags.

[055] Fig. 8-11 illustrates four exemplary embodiments of the floating point unit 100 with the different embodiments of the operand analysis circuit 120, the processing circuit 130, and the result generator 140 combined in various combinations. The floating point unit of Fig. 8 may be used when the floating point operand 200 stored in the operand buffer 110 may be in an zero, underflow, overflow, infinity, or NaN format according to U.S. Patent Application Serial No. 10/035,747 or in a normalized delimited format according to U.S. Patent No. 6,131,106. The floating point unit 100 of Fig. 9 may be used when the floating point operand 200 stored in the operand buffer 110 may be in

an overflow, infinity, or NaN format according to U.S. Patent Application Serial No. 10/035,747 or in a denormalized format according to IEEE 754. The floating point unit of Fig. 10 may be used when the floating point operand 200 stored in the operand buffer 110 may be represented according to IEEE Std. 754 or in a normalized delimited format according to U.S. Patent No. 6,131,106. The floating point unit of Fig. 11 may be used when the floating point operand 200 stored in the operand buffer 110 may be represented according to IEEE Std. 754 including the denormalized format according to IEEE 754.

[056] In summary, floating point unit 100 may be used to compute the high part of a floating point operand. The sign of the result generated by the floating point unit 100 may be the same as the sign 210 of the floating point operand 200 stored in the operand buffer 110. If the operand 200 is in a NaN format according to U.S. Patent Application Serial No. 10/035,747 or IEEE Std. 754, then the result generated by the floating point unit 100 may be the same NaN format (i.e., no bits are cleared). If the operand 200 is in an infinity format according to U.S. Patent Application Serial No. 10/035,747 or IEEE Std. 754 or an overflow format 1240, then the result generated by the floating point unit 100 may be a binary zero with the same sign as the operand 200. If the operand 200 is in a zero format according to U.S. Patent Application Serial No. 10/035,747 or IEEE Std. 754, then the result generated by the floating point unit 100 may be in a zero format with the same sign as the operand 200. If the operand 200 is a subprecise number, then the sign bit, the exponent field bits, and the fraction field high part bits of the result may be the same as the sign bit 210, the exponent field bits 220, and the fraction field high part bits 230 of the of the operand 200, respectively. The

resulting fraction field low part bits are adaptively determined based on the fraction field low part bits 240 and/or the fraction field high part bits 230 of the operand 200. If the operand 200 is in any other format, then the sign bit, the exponent field bits, and the fraction field high part bits of the result may be the same as the sign bit 210, the exponent field bits 220, and the fraction field high part bits 230 of the operand 200, respectively. The resulting fraction field low part bits may be cleared.

[060] Still further, the above description of the floating point unit 100 has been limited to operands according to U.S. Patent Application Serial No. 10/035,747, filed on even date herewith in the name of Guy L. Steele Jr. and entitled "Floating Point System That Represents Status Flag Information Within A Floating Point Operand," assigned to the assignee of the present application, U.S. Patent No. 6,131,106, and IEEE Std. 754. However, the floating point unit may be adapted to receive a floating point operand having a different format including having a different bit positions for storing status information. Adapting the floating point unit to receive a floating point operand having a different format will be obvious to those of ordinary skill in the art.

[061] Finally, the floating point unit 100 and all or part of its functionality may be implemented in software, firmware, hardware, or any combination thereof. For example, the invention may be practiced in an electrical circuit comprising discrete electronic elements, packaged or integrated electronic chips containing logic gates, a circuit utilizing a microprocessor, or on a single chip containing electronic elements or microprocessors. It may also be provided using other technologies capable of performing logical operations such as, for example, AND, OR, and NOT, including but not limited to mechanical, optical, fluidic, and quantum technologies. Any program may

in whole or in part comprise part of or be stored on the system in a conventional manner, or it may in whole or in part be provided in to the system over a network or other mechanism for transferring information in a conventional manner. In addition, it will be appreciated that the system may be operated and/or otherwise controlled by means of information provided by an operator using operator input elements which may be connected directly to the system or which may transfer the information to the system over a network or other mechanism for transferring information in a conventional manner.

[062] Other embodiments of the invention will be apparent to those skilled in the art from consideration of the specification and practice of the invention disclosed herein. For example, the floating point unit 100 may be implemented in software, firmware, hardware, or any combination thereof. It is intended that the specification and examples be considered as exemplary only, with a true scope and spirit of the invention being indicated by the following claims.